Reducing Power Consumption in Backbone Networks

Reducing Power Consumption in Backbone Networks

Luca Chiaraviglio, Marco Mellia, Fabio Neri
Dip. di Elettronica, Politecnico di Torino, Italy, Email: {last name}@tlc.polito.it

***Abstract*—According to several studies, the power consumption of the Internet accounts for up to 10% of the worldwide energy consumption, and several initiatives are being put into place to reduce the power consumption of the ICT sector in general. To this goal, we propose a novel approach to switch off network nodes and links while still guaranteeing full connectivity and maximum link utilization. After showing that the problem falls in the class of capacitated multi-commodity flow problems, and therefore it is NP-complete, we propose some heuristic algorithms to solve it. Simulation results in a realistic scenario show that it is possible to reduce the number of links and nodes currently used by up to 30% and 50% respectively during off-peak hours, while offering the same service quality.**

Keywords: green networks, power-aware, optimization

I. Introduction

Power consumption in general, and of ICT technologies in particular, has become a key issue during the last few years. The ratio of power demand versus power resources is constantly growing, and energy cost is increasing at a constant rate. Electricity costs jumped up of about 35% in Italy during the period 2004-2007 [1]. Moreover, Green House Gases (GHG) emissions have a negative impact on the world climate [2], and people are becoming more conscious about environment protection issues.

According to a number of studies, ICT alone is responsible for a percentage which varies from 2% to 10% of the world power consumption [3], due to the ever increasing diffusion of electronic devices. In this scenario, the power consumption of telecommunication networks, and of the Internet in particular, is not negligible. For example, considering a data center, the network infrastructure alone is responsible, according to [4], of 23% of the overall power consumption, even without taking into account the energy necessary for equipment cooling.

The study of power-saving network devices has been introduced over these years, starting from the pioneering work of [5]. In [6] the ideas of Adaptive Link Rate (ALR) and protocol proxying are proposed. Both these techniques require to change protocols, and both consider pairs of devices, e.g. routers sharing the same link, or a couple of high/low performance servers.

More recently, some effort was devoted to investigate how to reduce the power consumption of the entire network infrastructure, and not of single or few components only. In [7] some simple measurements about power consumption of networking devices are first presented; then authors consider a network topology and evaluate the total network consumption given the power footprint of each element. They consider two scenarios: in the first one, all devices are turned on, while in the second one only the minimum number of elements to guarantee the service are actually powered on. The reduction of the corresponding power consumption is finally evaluated.

In this paper, we consider a wide-area network scenario. Given a network topology and a traffic demand, we evaluate the possibility of turning off some elements (nodes and links) under connectivity and Quality of Service (QoS) constraints. The goal is to minimize the total power consumption of large networks, in which usually resource overprovisioning is large due to traffic dynamics and to the deployment of redundant resources for fault protection. We investigate some simple optimization algorithms. In particular, we selectively power off nodes and links of the topology following different strategies. Results show that it is possible to reduce the percentage of nodes and links actually powered on to up 30% and 50% respectively, while guaranteeing that the resource utilization is still below a given threshold. e.g., 50%.

We recognize that this work is somehow preliminary, since a careful evaluation of the network devices to be switched off cannot ignore how to deal with failures or considering the signalling overheads added to the system. Nevertheless our study permits to estimate the possible energy savings, obtained with simple heuristics, still guaranteeing QoS for users.

II. Problem Formulation

An informal description of the design problem we consider is the following: **Given** i) a physical network topology comprising routers and links, in which links have a known capacity, ii) the knowledge of the average amount of traffic exchanged by any source/destination node pair, iii) the power consumption of each link and node, **Find** the set of routers and links that must be powered on so that the total power consumption is minimized, **Subject to** flow conservation and maximum link utilization constraints.

More formally, we can provide an Integer Linear Programming (ILP) formulation of the problem to precisely define it. Let us represent the network infrastructure as a di-graph $G = (V, E)$, where V is the set of vertices and E is the set of edges. Vertices represent network nodes, while edges represent network links, being $N = \mid V \mid$ and $L = \mid E \mid$ the total number of nodes and links respectively. Let c_{ij} be the capacity of the link from node i to node j and let $\alpha \in \{0, 1\}$ be the maximum link utilization that can be tolerated [1]. Let t^{sd} be

[1]Link utilization is normally kept below 100% due to QoS requirements.

the average amount of traffic going from node $s = 1, \ldots, N$ to node $d = 1, \ldots, N$, i.e., $\{t^{sd}\}$ represents the traffic demand.

Let $x_{ij} \in \{0, 1\},\ i = 1, \ldots, N,\ j = 1, \ldots, N$ be binary variables that take the value of 1 if link from node i to node j, i.e., (i, j), is present and powered on. Similarly, let $y_i \in \{0, 1\},\ i = 1, \ldots, N$ be binary variables that take the value of 1 if node i is powered on. Let $f_{ij}^{sd} \in [0, t^{sd}]$ denote the amount of flow from s to d that is routed through the arc from i to j. Similarly, let f_{ij} be the total amount of traffic flowing on the link from i to j.

Finally, let $\mathcal{PL}_{ij}$ and $\mathcal{PN}_i$ be the power consumption of link from i to j, and of node i, respectively.

Given the previous definitions, it is possible to formalize the problem as follow:

Minimize

$$\mathcal{P}_{tot} = \sum_{i=1}^{N}\sum_{j=1}^{N} x_{ij}\mathcal{PL}_{ij} + \sum_{i=1}^{N} y_i \mathcal{PN}_i \tag{1}$$

Subject to:

$$\sum_{j=1}^{N} f_{ij}^{sd} - \sum_{j=1}^{N} f_{ji}^{sd} = \begin{cases} t^{sd}, & \forall s, d, i = s \\ -t^{sd}, & \forall s, d, i = d \\ 0, & \forall s, d, i \neq s, d \end{cases} \tag{2}$$

$$f_{ij} = \sum_{s=1}^{N}\sum_{d=1}^{N} f_{ij}^{sd} \quad \forall i, j \tag{3}$$

$$f_{ij} \leq \alpha c_{ij} x_{ij} \quad \forall i, j \tag{4}$$

$$\sum_{j=1}^{N} x_{ij} + \sum_{j=1}^{N} x_{ji} \leq M y_i \quad \forall i \tag{5}$$

Eq. (2) states the classical flow conservation constraints, while Eq. (3) evaluates the total flow routed on each link. Constraint (4) forces the total link offered load to be smaller than α, while constraint (5) states that a node can be turned off only if all incoming and outgoing links are actually turned off. The big-M method is used to force this constraint, $M \geq 2N$.

Notice that according to Eq. (2) routing is assumed to be optimal. Since f_{ij}^{sd} is a real variable, flows are routed using a fluid model.

The presented formulation falls in the class of capacitated multi-commodity minimum cost flow problems (CMCF) [8], i.e., the problem in which multiple commodities have to be routed over a graph with capacity constraints. CMCF problems are known to be NP-hard, so exact methods can only used to solve trivial cases. In this paper, we therefore propose some simple heuristics in order to solve the design problem for large networks. Moreover, since $\mathcal{PL}$ and $\mathcal{PN}$ are difficult to know and vary widely depending on the considered technology, in the following we consider for all the devices $\mathcal{PN} = 1$ and $\mathcal{PL} \ll \mathcal{PN}$ [6], so that the objective function can be pursued by trying to switch off the largest possible number of network nodes.

```
//node optimization
sort_nodes(node_array, order_type);
for (i=0; i<N; i++) {
    disable_node(node_array[i]);
    //for each (s,d) pair, compute the
    //shortest path. In case of tie,
    //pick one shortest path at random
    paths=compute_all_shortest_path();
    compute_all_link_flow(paths);
    if (check_paths(paths) == false)||
                (check_flows(paths) == false))
                enable_node(node_array[i]);
}
//link optimization
sort_links(link_array, order_type);
for (j=0; j<L; j++) {
    disable_link(link_array[j]);
    paths=compute_all_shortest_path();
    compute_all_link_flow(paths);
    if (check_paths(paths) == false)||
                (check_flows(paths) == false))
                enable_link(link_array[j]);
}
```

Listing 1. Pseudo-code description of the proposed algorithms.

III. Algorithms

The heuristics we propose start by considering a network in which all elements are powered on, so that $x_{ij} = 1\ \forall i, j$ and $y_i = 1\ \forall i$. Each algorithm then iteratively tries to switch off an additional network element (either a node or a link).

At each step, traffic flowing through the network element to be switched off is rerouted on the shortest path for each (s, d) pair [2] to verify Eq. (2), and the utilization constraint (4) is checked for all links. If no violation is present, then the selected element is powered off. Listing 1 reports a schematic description of the algorithms.

Algorithms we present in this paper share the same intuition: the energy saving acheived by turning off nodes is higher than by switching off single links [6], and switching off a node is more difficult than switching off a single link. Algorithms therefore try to turn off nodes first, and then links are possibly powered off in a second iteration.

Several policies can be adopted to iterate through the node set. We implemented the following ones: random (R), least-link (LL), least-flow (LF), opt-edge (OE).

The node set is first sorted considering a given rule before iterating through all the nodes. The random heuristic sorts nodes in random order. The least-link heuristic sorts the nodes according to the number of links that are sourced and sinked at each node, so that nodes with a smaller number of links are checked first, i.e., V is sorted in increasing value of

$$X_i = \sum_{j=1}^{N} x_{ij} + \sum_{j=1}^{N} x_{ji}.$$

[2] In this phase, differently from the ILP formulation, we use a simple shortest path algorithm to route the traffic. Moreover, f_{ij}^{sd} is supposed to be an integer variable, so that $f_{ij}^{sd} \in \{0, t^{sd}\}$.

```
//for each edge node not yet visited
void opt_edge(NODE curr_n, NODE node_array[]) {
  //node_array is initially empty; nodes to be
  //switched off are progressively inserted
  edge_visited[curr_n]=can_be_off=true;
  edge_off[curr_n]=false;
  //PHASE 1: check if the neighbors are on.
  //Get the list neighbor_n of edge nodes that
  //are reachable from curr_n using all
  //aggregation nodes connected to curr_n
  neighbor_n=find_neighbors(curr_n);
  //the current node can be off only if all
  //neighbors are on
  while(neighbor_n!=NULL) {
    nn=neighbor_n->id_n;
    if(!visited[nn])) {
      edge_visited[nn]=true;
      edge_off[nn]=false;
      //update the list of neighbors that
      //have to be checked later on
      insert_in_list(nn,list_on);
    }
    else {
      if(edge_off[nn]) {
        can_be_off=false; break;
      }
    }
    neighbor_n=neighbor_n->next;
  }
  //PHASE 2: turn off the current edge node.
  if(can_be_off) {
    //all the edge neighbors are on, then
    //put curr_n in the top of node_array
    edge_off[curr_n]=true;
    insert_into_array(node_array,curr_n);
  }
  //PHASE 3: go to the second grade neighbors.
  if(list_on!=NULL) {
    //call opt_edge for second grade neighbors
    for(i=0; i<list_on.size; i++) {
      neighbor_n=find_neighbors(list_on->id_n);
      while(neighbor_n!=NULL) {
        nn=neighbor_n->id_n;
        if(!edge_visited[nn])
          opt_edge(nn, node_array);
        neighbor_n=neighbor_n->next;
      }
    }
  }
}
```

Listing 2. Pseudo-code description of the opt-edge algorithm.

The least-flow heuristic takes instead into account first the nodes with the smallest amount of information flowing through them, i.e., V is sorted in increasing value of

$$F_i = \sum_{j=1}^{N} f_{ij} + \sum_{j=1}^{N} f_{ji}.$$

Finally, the opt-edge algorithm relies on the fact that in actual Internet topologies, user traffic is collected by means of "aggregation" nodes, e.g., DSLAMs or corporate networks gateway nodes. For protection purpose, these nodes are typically multi-homed, i.e., they are connected to two "edge" nodes that collect traffic from several aggregation nodes in the same area. Aggregation nodes are traffic sources and sinks, so that they cannot be powered off. Similarly, only one of the two edge nodes can possibly be powered off, while the other one must be active. The opt-edge algorithm exploits this, by walking through the lists of edge nodes and aggregation nodes to extract the list of nodes that can be powered off without violating this constraint.

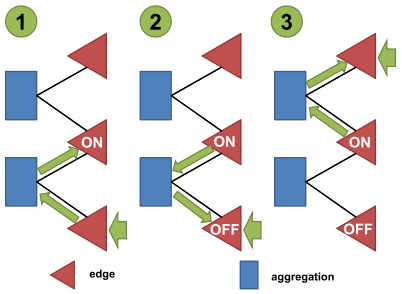

Fig. 1. Scheme of the opt-edge algorithm.

Listing 2 reports a schematic description of the opt-edge algorithm, which is shown also in Fig. 1. The key point is to start from an edge node and to find all the edge neighbors still powered on by going through aggregation nodes. If all its neighbors are on, then the current node can be powered off, and it is inserted in the corresponding vector. Recursion is exploited to explore the neighborhood from the current node, so that the function is invoked until all edge nodes are visited. The function opt-edge orderly inserts the nodes that can be switched off in node-array, which is later completed with the other nodes to implement the sort node function of Listing 2.

Considering algorithm to turn off links, two sorting criteria are considered: least-flow (LF), random (R). Both of them leverage on the same intuition than the corresponding node sorting heuristics: the least-flow policy sorts links in increasing order of carried flow, i.e., E is sorted in increasing value of f_{ij}, while the random policy sorts links in random order.

All possible node/link sorting combinations have been studied. Besides these heuristics, we also tested the corresponding ones in which a decreasing order is adopted. Since they all perform consistently worse (as expected), we decided not to consider them in this paper.

IV. Performance Comparison

In order to assess the performance of the proposed algorithms, we consider a wide-area network scenario. The goal is to show that, for a given traffic demand, it is possible to power off some network elements, and to still guarantee full connectivity between sources and destinations, while enforcing that the link utilization remains below a QoS threshold. Obviously this depends on the network topology and traffic demand. With more redundant resources in the network, it is easier to save power consumption. We consider a scenario inspired by the real network of an ISP, in which redundancy is mainly related to fault protection requirements.

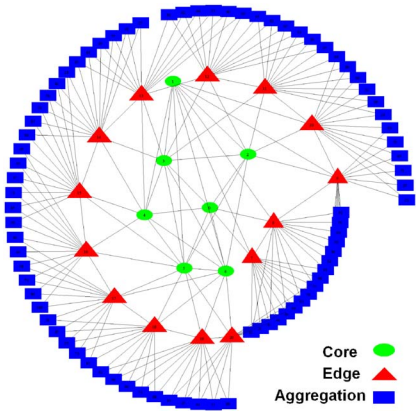

Fig. 2. An example of random topology.

We suppose a hierarchical topology, which is typical of WANs. All links are supposed to be bidirectional links, so that if link (i, j) exists, then link (j, i) exists as well. Three levels of nodes are considered: core, edge and aggregation nodes.

The network core is composed by few nodes that are highly interconnected by means of high-capacity links. Each link connects nodes which may be also geographically far away, e.g., optical links connecting different cities.

The edge nodes are used to interconnect aggregation nodes to the core nodes. Links between edge and core nodes have middle-range capacity, i.e., smaller capacity than the links interconnecting core nodes. Each edge node is connected to some of the closest core nodes, and to other edge nodes. One or more edge nodes can be present in cities, and they collect traffic from aggregation nodes spread within the city boundaries.

The last level of nodes is composed by the aggregation nodes, to which users are directly connected. A Digital Subscriber Line Access Multiplexer (DSLAM) or an Optical Line Termination (OLT) in PONs are typical examples of aggregation node. Each node is dual-homed, i.e., it is connected to the closest pair of edge nodes (to guarantee alternate paths in case of failure). Low capacity links connect aggregation nodes to edge nodes.

In building the topologies used to test our algorithms, for a given number of nodes of the different types, network links are randomly positioned between nodes.

Results have been obtained considering randomly generated topologies in which 160 nodes are considered. In particular, 10 core nodes, 30 edge nodes, and 120 aggregation nodes are considered. Nodes are assumed to be placed on a plane. Core nodes are randomly connected to other core nodes with probability $p = 0.5$. Each edge node is then connected to the two closest core nodes and to another randomly selected edge node. Finally, aggregation nodes are connected to the two closest edge nodes. An example of the topology obtained is presented in Figure 2. Aggregation, edge and core nodes are represented by squares, triangles and circles respectively.

Only aggregation nodes are traffic sources and sinks. For the sake of simplicity, we consider a uniform traffic pattern, so that $t^{sd} = U[0.5, 1.5]$ units of traffic if s and d are aggregation nodes; $t^{sd} = 0$ otherwise.

Given nodes and links, links capacities are selected according to the following approach. Three classes of links are defined: high, middle-range and low capacity links. Each link class has a minimum capacity c_{ij}^{min}, that was selected to be 15, 5, and 1 units of traffic respectively. Minimum link capacities will also be used as link routing weights, so that the routing cost is inversely proportional to the link capacity. This is commonly adopted to force the traffic to be routed through the edge and the core nodes, rather than through aggregation nodes (which are connected by means of low capacity links). A simple minimum-cost path is considered as routing algorithm, similarly to what is commonly adopted in the Internet. Given the traffic matrix and the routing, link flows can be computed, and a different capacity is assigned to each link, possibly in excess to the minimum capacity, so that the total traffic flowing through a link is forced to be smaller than an overprovisioning factor $\beta = 0.5$. Therefore, after routing all traffic, link capacities are assigned so that:

$$c_{ij} = \max(\lceil f_{ij}/\beta \rceil, c_{ij}^{min})$$

A. Heuristic Comparison

For each considered heuristics, we collected the percentage of nodes and links that are turned off, η_N and η_L respectively. This test was repeated on 20 randomly generated topologies and traffic patterns. Fig. 3 and Fig. 4 show the comparison of the different heuristics by reporting η_N and η_L respectively. Bars report mean values, while the error bars show the standard deviation. Labels on the x-axis report the node-link heuristic combination. A maximum link load factor $\alpha = 0.5$ was considered. Experiments consider a off-peak hour traffic scenario $t^*_{s,d}$, in which traffic is the 20% of the peak demand ($t^*_{s,d} = 0.2\, t_{s,d}$).

We report an upper-bound (dashed line) obtained by relaxing constraint (4), so that only the flow conservation constraint is imposed. This is equivalent to find the minimum set of nodes and links that permit to route all the offered traffic, and it allows to better assess the impact of the QoS constraint, and the quality of the solutions generated by the proposed heuristics.

Fig. 3 reports the average number of nodes that different heuristics are able to switch off: in this case, the OE-LF and OE-R heuristics are very close to the optimal solution. All the other algorithms perform consistently worse, by leaving in power on state 5% to 8% of additional nodes. Notice also the smaller standard deviation exhibited by the OE algorithms, which proves the effectiveness of the OE heuristics.

Considering η_L (Fig. 4), we can see that it is possible to actually turn off about 30% of links in the considered network and traffic scenario. The best performance is obtained by the OE-LF and OE-R algorithms, even if other algorithms show very similar results. All of the heuristics are in fact quite close to the upper-bound, which means that little improvement is possible in our scenario.

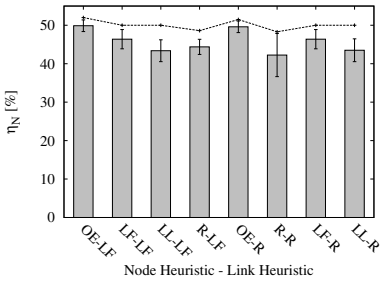

Fig. 3. Comparison of the percentage of nodes switched off considering different algorithms. Off-Peak traffic scenario.

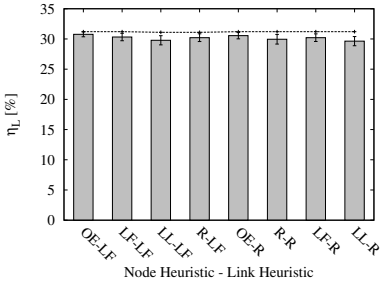

Fig. 4. Comparison of the percentage of links switched off considering different algorithms. Off-Peak traffic scenario.

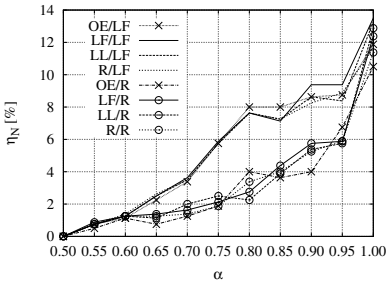

Fig. 5. Percentage of nodes switched off versus α.

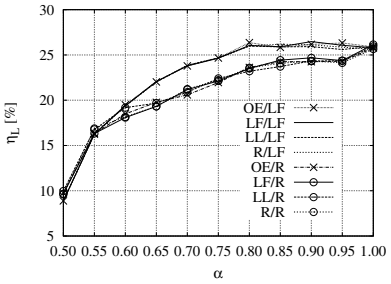

Fig. 6. Percentage of links switched off versus α.

B. Parameter Impact

We performed a study on the impact of the α parameter [3] to observe the range of network elements that can be successfully switched off by increasing the maximum offered load links can sustain. The rational of changing α is to analyze the behaviour of our approach as a function of the amount of spare resources (network capacity versus traffic demand). Peak-hour traffic is considered. For sake of simplicity, only mean values are reported for each heuristic combination.

Fig. 5 shows that a larger number of nodes can actually be switched off for larger value of α. Little impact is shown by the node sorting criterion, while the LF link sorting heuristic consistently performs better than a random order. Notice however that only up to 13% of nodes can be powered off considering a maximum link load factor $\alpha = 1$, which is clearly infeasible because of traffic congestion.

Fig. 6 reports the number of links switched off for $\alpha \in [0.5, 1]$. All algorithms show large improvements for α up to 0.8; after that, little improvement is noticeable, and a final minor decrease in the percentage of links that can be turned off is observed for values of $\alpha > 0.8$. This is caused by the corresponding increased number of nodes turned off, which reduces the freedom of turning off links, since not many alternate paths remain available.

In the following, we investigate the effect of load traffic variation with time. Actual traffic is known to change according to a day-night trend. Here we assume that traffic load changes according to a sinusoidal function, with daily periodicity, i.e:

$$t^{sd}(t) = t^{sd}\left[\frac{1-\gamma}{2}\left(1+\sin\left(f_o t\right)\right)+\gamma\right] \qquad (6)$$

where $f_o = 1/24$ hours. We assume that during night the mean amount of traffic is equal to the 20% of the peak traffic, so that $\gamma = 0.2$.

Traffic is supposed to be uniformly distributed [4] among aggregation nodes only, and routing weights and link capacities are assigned as previously described. LF-LF and OE-LF algorithm are assessed, considering a scenario in which $\alpha = 0.5$, i.e., with the same QoS constraint enforced during design.

Fig. 7 shows the variation of the percentage of nodes switched off versus time. Interestingly, during night time the percentage of nodes switched off by the algorithms allows to achieve large savings. In particular, the OE-LF algorithm exhibits performance very close to the upper-bound (50%), while the LF-LF algorithm shows consistently worse results (45%). On the contrary, during the day, the percentage of nodes switched off is equal to zero, since the whole network capacity is required to satisfy the traffic demand.

Considering the percentage of links that can be switched off (Fig. 8), we can notice that even during the day, it is possible to turn off some links, e.g., one out of two links between the aggregation and edge nodes that are installed for protection purpose, and therefore do not carry traffic in normal conditions.

[3]Notice that the overprovisioning factor β is kept equal to 0.5 when the capacities c_{ij} are assigned. During optimization instead the overprovisioning factor α ranges between 0.5 and 1.0.

[4]Real traffic models are more bursty, but for a first estimation of possible savings a uniform model is enough.

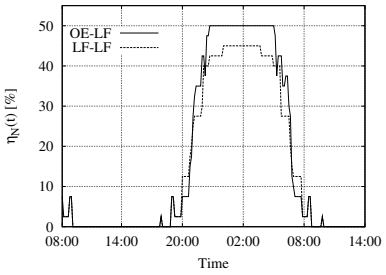

Fig. 7. Percentage variation of nodes switched off versus time. The LF-LF and OE-LF algorithm are considered.

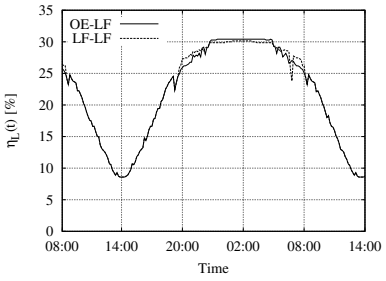

Fig. 8. Percentage variation of links switched off versus time. The LF-LF and OE-LF algorithms are considered.

The last part of our work studies the impact of the network size on the capability of switching off some elements. Different network topologies were generated, with increasing number of nodes N. In particular, the relative ratio between core, edge and aggregation node is constant, so that x nodes are core nodes, $3x$ are edge nodes, and $12x$ are aggregation nodes ($N = x + 3x + 12x$). The aim is indeed to generate topologies that have similar characteristics, but different number of nodes.

Traffic is supposed to be uniformly distributed among aggregation nodes only, considering the off-peak hour scenario. Routing weights and capacity assignment are performed as previously described. The OE-LF algorithm is tested to observe the percentage of nodes and links that can be switched off, considering $\alpha = 0.5$. Results are averaged over 5 different topologies.

Fig. 9 shows on the right the variation of the percentage of links switched off for increasing values of x. Interestingly, η_L is independent from x; indeed, about 30% of links are switched off for all the considered networks.

The left part of Fig. 9 shows instead the results considering η_N. In this case, larger values of x lead to smaller values of η_N; indeed the percentage of nodes that are switched off decreases from 53% for $x = 6$ to 46% for $x = 30$. The intuition suggests that the distance between nodes increases as $O(\log N)$, so that the amount of traffic that is rerouted on alternate paths when a node is switched off increases the offered load on several links and nodes, making it difficult to completely switch off nodes.

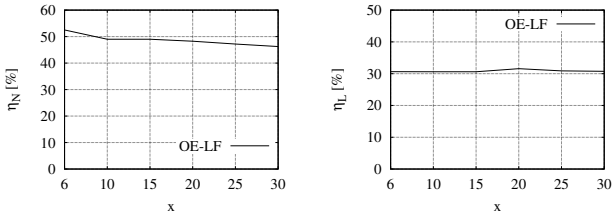

Fig. 9. Percentage variation of nodes (left) and links (right) switched off versus different values of x. The OE-LF algorithm is considered.

V. Conclusions and Future Work

In this paper we faced a network design problem. We deviated from the traditional formulations of the problem, in which the objective function is to minimize cost or maximize performance, by considering as objective function the minimization of the total power consumed by the network, while connectivity and maximum link utilization are taken as constraints.

We provided an integer linear programming formulation of the problem, which shows that it is a NP-complete problem. Greedy heuristics have been proposed, and their performance assessed considering some simple yet realistic traffic and network scenarios. Results (although dependent in absolute values from the chosen scenario) show that it is possible to switch off both nodes and links, so that the total network power consumption can be reduced. In particular, during off-peak hour, traffic is much smaller, so that is possible to reroute it on a subset of network resources and switch off a large number of nodes and links.

Acknowledgment

The work described in this paper was performed with the support of the BONE project (Building the Future Optical Network in Europe), funded by the European Commission through the 7th Framework Programme, and the WiFi4Energy project, funded by Regione Piemonte.

References

[1] *Eurostat Web Page*, http://epp.eurostat.ec.europa.eu/.
[2] Katalin Szomolnyi, "GreenHouse Gas Effect of Information and Communication Technologies", *ETNO Project Study*, http://www.etno.be/, 2005.
[3] "An inefficient truth", *Global Action Plan Report*, http://www.globalactionplan.org.uk/, December 2007.
[4] "Where Does Power Go?", *http://www.greendataproject.org/*, January 2008.
[5] M. Gupta, S. Singh, "Greening of the Internet", *Proceedings of ACM SIGCOMM*, Karlsruhe, Germany, August 2003.
[6] K. Christensen, C. Gunaratne, B. Nordman, "Managing Energy Consumption Costs in Desktop PCs and LAN Switches with Proxying, Split TCP Connections, and Scaling of Link Speed", *International Journal of Network Management*, Vol. 15, No. 5, pp. 297-310, September 2005
[7] P. Barford, J. Chabarek, C. Estan, J. Sommers, D. Tsiang, S. Wright, "Power Awareness in Network Design and Routing", *IEEE INFOCOM 2008*, Phoenix, USA, April 2008.
[8] T. G. Crainic, M. Gendreau, I. Ghamlouche, "Cycle-based neighbourhoods for fixed-charge capacitated multicommodity network design", *Technical report CRT-2001-01*, Centre de recherche sur les transports, Universite de Montreal, 855.